Apostle: 協調結合に基づく分散処理システム(1)
—— Trinity 構想における構成 ——

井田昌之，田中啓介
青山学院大学情報科学研究センター

1 Trinity 構想

Trinity 構想は，青山学院大学情報科学研究センターで進められている3キャンパスネットワーク構想である．1988年4月に設置された研究教育開発室を中心に実験が開始されている．Apostle はその実験システムの名である．協調結合[1]はその際に用いられるネットワーク概念である．

Trinity 構想は，次の枠組みを持っている．
1. 広域性を持つ：距離的に離れた3キャンパス(青山，世田谷，厚木)間の計算機設備を接続したネットワークを構築する．
2. 異機種性を持つ：スーパーコンピュータからパソコンにいたる諸設備の効果的な運用の一貫として，それらを，ネットワーク化する．
3. 汎ネットワーク性を持つ：学外のネットワークへのゲートウェイ機能，学内各部の異なるアプリケーション層を持つ LAN に対する幹線としての機能を提供する．

構想実現に当たって，設計上のポイントとして次の点を加味する必要がある．
A. 専任教員(助手を含む)のおよそ30%は，2キャンパス以上にまたがる職務を持つ．また，そのほとんどは計算機科学の専門家ではない．従って，どのキャンパスからも同一環境に，同一の手順，しかも極力簡単な操作でアクセスできることが望ましい．
B. 研究者用ネットワークである．
C. 青山・世田谷間が約10km，青山・厚木間が約45kmであり，それらの間に，例えば Ethernet のようなバス結合の回線を持つことは，効果・コスト比の点から困難である．
D. 電子メール，ニュースシステム，及び，データベースの共通アクセス等の学内情報交換の手段がほしい．

2 協調結合

分散環境に対応するための既存の技術としては，
1)(密結合型の)分散 OS，例えば Mach[2]，を用いて作業環境を提供する，
2)NFS[3]などの分散ファイルシステムを用いて，統合ファイル環境を提供する，

表 1: 協調結合の位置

	(密結合)分散OS	協調結合	分散ファイルシステム
例	Mach	*Apostle*	NFS+YP
結合媒体	バス Ethernet	同期回線 X.25	Ethernet
traffic	大	小	中
環境の同一性	同一	論理的に同一	共通
分散度	小	中	大

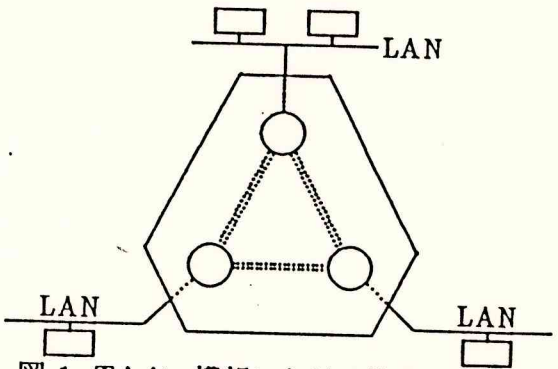

図 1: Trinity 構想における協調結合モデル

などをあげることができる．しかし，これらは，いずれも高速な回線(専用バスもしくは Ethernet)を用いた結合を前提とし，本構想の条件に合わない．一方，機能的には，1)が提供する同一環境，2)が提供する軽便さ・既存 OS(4BSD)との親和性は確保したい．そこで，表1のような位置づけを持つ体系として，協調結合と呼ぶ新しい概念の確立を試みた．協調結合は次のような概念を持った，疎結合型分散 OS 概念である．
「遠距離にあるコンピューターを構成要素とする分散 OS．それらの間の結合は，任意の媒体，例えば，同期専用回線，DDX-P，公衆回線を可とする．」このように実現することにより，同一環境の提供を特別な高速回線によらずに，実効的な密結合を達成できることを意図している．図1にそのモデルを示す．

Apostle: A Distributed System based on the Harmonic Connection (1) -Its overall design in the Trinity Initiative -
Masayuki IDA, Keisuke TANAKA
Aoyama Gakuin University

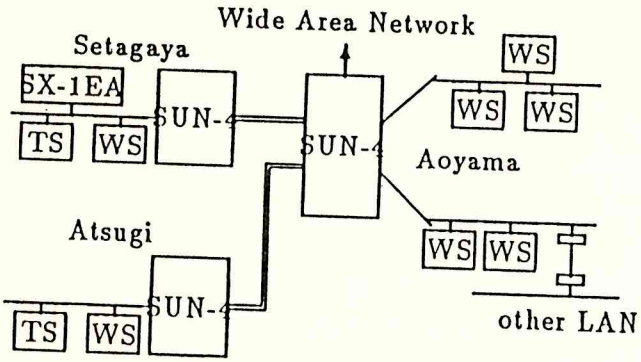

図 2: Apostle の構成

```
4.2 BSD UNIX (Apostle)

Atsugi login: m-ida
password:
Last login: Tue Jun 14 11:05:01 from backabon
atsugi% cd x3j13; pwd
/home/aoyama/m-ida/x3j13
atsugi% ls
cl.mbox        clos.mbox      senddoc.1
cl-edit.mbox   packmail       x3j13.mbox
atsugi% packmail cl.mbox
atsugi% ex /tmp/work.doc
"work.doc" 213/4378
:r ~keisuke/important.doc
access denied
"important.doc" No such file or directory
:!ls /usr/spool
access denied
/usr/spool not found
!
:q!
atsugi% logout
```

図 3: 会話例(青山の home を厚木から使用)

3 協調結合の実現

図 2 に Apostle の構成を示す. 3 キャンパスにあるゲートウェイを 3 要素とする分散 OS として Apostle を構成する. Apostle を構成する 3 コンピュータは運用のしやすさから同一計算機とし, それらに対して, 各キャンパスの LAN を接続する. 具体的には, 次のような構成とした.

1)SUN4 (SunOS 4.0 以降) を用いる.
2)YP[4], Internetwork Router[5] を基盤とする.
3)接続は, NTT スーパーディジタル回線を用いる. 回線の制御は, MMM (NEC) を用いる. 64kbps を設計速度とする.

基本的な構造を次のようにまとめた.

1) Information Server, User Interface, File Cacher の 3 部により構成する.

2)Information Server は, 利用者に関する情報を管理する. 各キャンパスの Component Computer には同一の情報のコピーが置かれる. YP に対して, 新たにサブシステムを作成することにより YP の拡張として実現する.

3)User Interface は, このために開発する分散型の restricted shell と UNIX の system call の拡張である. 保護された同一の環境を, どのキャンパスからのアクセスに対しても提供する.

4)File Cacher は, 利用者がアクセスしたファイルが, ログインした Component と異なる Component(遠隔地) にある場合のキャッシングを行なう. ファイルのアクセス権は 4 レベルに分けて管理する. Private, Group, Public, System である. これらの詳細は[6]に述べる.

利用者に関しては, 一般利用者の特質から,「同一利用者が, 同時に別のキャンパスからログインする事はない.」ということを前提としている.

4 現状と今後

Apostle の第一版は 1989 年 3 月稼働を目指している. また, Trinity 構想全体は 3 年間の計画である. 現在, パイロットモデルを研究室内において実現し, SUN4 及び, 結合形式に対する評価を行なっている. 現状では即断できないが, それによれば, 一般利用者は, 独自のプログラム開発は少なく, 多くが, UNIX 自身及び, 運用者側で提供した機能 (Public 及び System file の実行アクセス) を用いてプライベートファイルを更新する操作であることが解析されている. Apostle は, この形態に適したシステムであると評価している. 図 3 に会話例を示す.

謝辞: 情報科学研究センター所長大矢知浩司教授並びに, 各副所長, センター諸氏に感謝致します.

References

[1] Masayuki Ida, Keisuke Tanaka, *'The Harmonic Connection Concept in the Trinity Initiative of Aoyama Gakuin University,'* Proc. of 3rd JCCW-88, July 1988.

[2] M.Accetta, et. al. *'Mach: A New Kernel Foundation for UNIX Development,'* in Proc. of USENIX 1986 summer conf. pp93-112, 1986.

[3] Sun Microsystems Inc. *'Network File System Protocol Specification,'* Feb. 1986

[4] -, *'The Yellow Pages Protocol Specification,'* Feb. 1986

[5] -, *'SunLink Internetwork Router System Administration Guide,'* Jul. 1987

[6] 田中啓介, 井田昌之, 'Apostle: 協調結合に基づく分散処理システム(2) -システムの実現-' 第 37 回情報処理学会全国大会, Sep. 1988.

Apostle: 協調結合に基づく分散処理システム(2) —— システムの実現 ——

田中啓介，井田昌之
青山学院大学情報科学研究センター

1 全体構成

Apostle は，現在，青山学院大学情報科学研究センターで進めている Trinity 構想の実現形態である[1]. *Apostle* は，3 キャンパスに設置される計算機(Component Computer)間結合を基盤とする．その主要な構成要素は，それぞれの Component Computer に実現される Information Server (IS), User Interface (UI), File Cacher (FC) の 3 つの機構であり，これらの機構により利用者に対してキャンパス間での統一的な計算機利用環境を提供する．図 1 に *Apostle* 全体の構成を示す．各構成要素は UNIX のプロセスとして実現され，機能の呼び出しはプロセス間通信機能を用いて実現される．

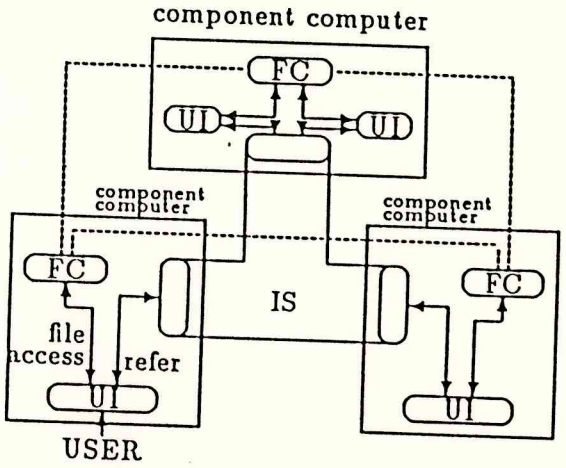

図 1: *Apostle* の全体構造

2 Information Server (IS)

IS は 3 キャンパスに分散したデータベース管理機構であり，三者が協調して一つのデータベースを管理する．データベースは，*Apostle* の動作に必要な利用者に関する情報を内容とする．

2.1 データベースの内容

データベースの内容は以下の通りである．

- *Apostle* 上での利用者認識名: 通常，課金処理に用いられる認識名は順番号に基づくため，直感的な利用者の認識には役立たない．従って，別の認識名を導入する．
- 管理者による課金処理用認識名
- 環境情報:
  統一的な利用環境を設定するために必要な個人情報であり，1) 個人の private file を保管するためのホームディレクトリ，2) メールボックス，3) 利用者が参加するグループに対してのホームディレクトリ，に関して物理的な位置(Component 名とパス名) が管理される．
- その他の個人情報:
  利用者の本名や所属部門，住所などが割り当てられる．これらの情報は Apostle システムそのもののためには用いられないが，今後開発されるシステム上のアプリケーションに対して提供される．

2.2 データの参照，更新

3 キャンパスの各 Component Computer において同一のデータが管理されることから，データの参照は各 Component Computer 内部で処理される．これによってデータ参照に対してキャンパス間の通信を行なう必要がなくなる．

しかし，常に各キャンパスでのデータベースの内容を同一に保つために更新の際に特殊な機構を必要とする．この更新動作は，以下の手順で行なわれる．

1. 更新要求の発生．この要求は Component Computer 内部で行なわれる．
2. 更新要求の正当性の確認．
3. 全ての Component Computer が動作していることを確認．動作状態にないものが存在する場合は，その回復を待つ．
4. 全ての Component Computer が動作している場合にのみ変更情報が伝えられ，データの更新が行なわれる．
5. 同時に互いに矛盾する変更要求が与えられた場合，その要求は失敗する．

この更新手順では，データベースの維持，管理に関するキャンパス間の通信量を最小にするため，一貫性の確認のような機能は提供しない．そのため，ある Component Computer が停止している状態では要求が待ち状態に入り直ちにデータベースに反映されることがないという欠点を持つ．

Apostle: A Distributed System based on the Harmonic Connection(2) -Implementation of *Apostle*-
Keisuke TANAKA, Masayuki IDA
Aoyama Gakuin University

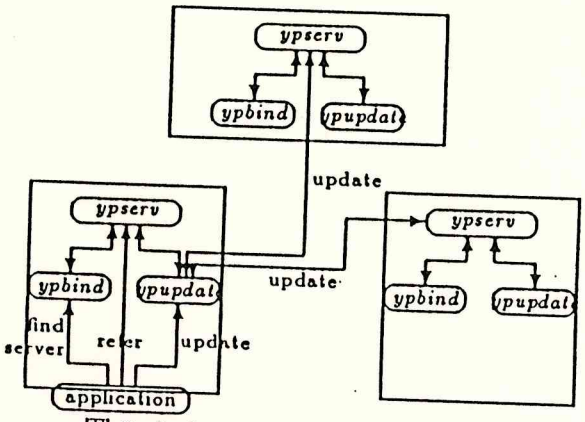

図 2: Information Server の構造

2.3 実現

IS は SUN Microsystems によって提供されている Yellow Pages (YP) システム[2]の拡張によって実現される. SUN が標準で提供する *ypserv(8)*, *ypbind(8)*に *Apostle* 用データを管理する上での変更を加えた上で, データの更新を専門に行なう *ypupdate* を新設した. これらの関係を図 2 に示す.

この実現方法においては, YP プロトコルそのものには変更を加えていない. これは *Apostle* システムの外部に存在する計算機の持つ機能との互換性を保証するためである.

3 User Interface (UI)

UI ではファイルタイプを以下の 4 種とする.

1. private file:
   各個人用のファイルであり, ある一つの Component Computer に置かれる. 基本的に所有者以外の利用者はアクセス出来ない.
2. group file:
   何人かの利用者で共有されるファイルであり, ある一つの Component Computer に置かれる. グループのメンバー以外の利用者はアクセス出来ない.
3. public file:
   一般コマンドやデータファイルであり, Component Computer 毎に置かれる.
4. system file:
   各 Component Computer のオペレーティングシステムが動作するために必要なファイル.

また, UI では利用者を 1)通常の利用者である一般利用者, 2)利用者登録や利用環境の設定等の一般的な管理作業を行なう通常管理者, 3)各 Component Computer の OS の正常動作のための管理を行なう Component 管理者, 4)全体を統括する *Apostle* システム管理者の四段階に分割する. 管理作業のため, 2) は管理範囲にある利用者の個人ファイルと管理上関係するファイル, 3) は system file, 4) は全てのファイルに対して特別に変更権が認められる. この管理権の分散によって, 各管理作業が明確になり, かつ個々の管理者の負担は減少する.

UI は UNIX の shell をフロントエンドとして実現される. また, アクセス権の制御のためにファイル操作に関する UNIX の system call が変更される. 利用者からのコマンドは shell によって解釈され, まず, その利用者がコマンドを実行可能か否かが判断され, 可能な場合にのみ実行が行なわれる. ファイル操作に関しての system call 内では利用者のファイルに対してのアクセス権がチェックされる. この際, 利用者についての情報を引き出すために IS の機能を利用する. また, アクセス対象が個人用のファイルであり, 他 Component Computer 上に存在する場合, そのファイルをアクセスするために File Cacher (FC) の機能を利用する.

4 File Cacher (FC)

利用者個人に対するファイルやグループに対するファイルは三キャンパス内の一つの Component Computer 上にのみ置かれる. したがって, 物理的に他の Component Computer にログインしている場合には, 個人用のファイルにアクセスするためにキャンパス間の通信が必要となる. この通信を最小化させる目的で FC が設置される.

利用者が個人用ファイルをアクセスする場合, その一回目のアクセスでその全内容が利用者が利用している Component Computer 上の一時使用領域に複写される. 以後の同一ファイルに対するアクセスは全てこの複写されたファイルに対して行なわれ, キャンパス間の通信は行なわれない.

ファイルを変更する目的でアクセスが行なわれた場合, 元のファイルには他の変更を拒否するようにロックがかけられる. 但し, private file に対しては原則的にファイルの所有者以外のアクセスは認められていないので, private file に対してのロックは省略される. 変更のあったファイルは利用者のログアウトの時点で元のファイルに書き戻される.

References

[1] 井田昌之, 田中啓介, 'Apostle: 協調結合に基づく分散処理システム(1) -Trinity 構想における構成-,' 第 37 回情報処理学会全国大会, Sep. 1988.

[2] Sun Microsystems Inc. *'The Yellow Pages Protocol Specification,'* Feb. 1986.